BİL 467/561 – Image Processing

Summer 2022

Final Project Report

|  |  |
| --- | --- |
| Your Name and Surname: | Hüseyin PEKKAN |
| Your Student ID: | 201401016 |
| Project Title: | Steganography with SIFT |
| Project Personnel: | Toygar AKGÜN |

1. Project Description and Background

* One paragraph: What is your goal? What is the problem you are trying to solve?
* A few paragraphs: What is the state-of-the-art regarding your technical goal? How are other people achieving the same goal? What are the most well-known, classic methods that have been used in the past and/or are still in use?
* One paragraph: A very brief and clear explanation of your work. Explain what you’ve done in two/three sentences. Don’t get too technical, just give a brief and crisp explanation.
* Benim amacım steganografi ile sifti birleştirerek halka açık alanda paylaşılan fotoğraflarda bir anahtar kullanarak bize gösterilmek istenen bölgenin çıkarılmasını sağlamaktı. Örneğin televizyonda halka açık bir şekilde yayınlanan fotoğraftan, bir devlet görevlisinin gerekli deşifreyi yapması ve fotoğrafta asıl gösterilmek istenen noktayı tespit etmesi.
* Steganografi kısmında şu an kullanılmakta olan yöntem, birleştirilmek istenilen fotoğrafların orijinal olanının en az anlamlı bitlerine (“Least significant bits”) gizlenmek istenilen fotoğrafın en anlamlı bitlerinin yerleştirilmesi ile orijinal fotoğrafta büyük bir kalite kaybı yaşamadan bir birleştirilme yapılması söz konusu. Fakat bunun SIFT ile birleştirilmiş bir versiyonunu ben göremedim. SIFT’in patenti Mart 2020’de bitti. Yani bu zamana kadar bir alternatifi mevcut değildi. Şu an ise bu amaçla daha ağırlıklı olarak CNN ile eğitilmiş modeller kullanılıyor.
* İlk önce elimizde şifrelenmiş bir fotoğraf olduğunu varsayalım, elimizdeki bu fotoğrafın her pikselindeki değeri 8 bit bir ikilik sayıya çeviriyoruz ve son 3 bitini alıyoruz ve buna 5 adet “0” ekleyerek gizlenmiş fotoğrafımızı elde ediyoruz. Bu işlemi yapmak isteyen kişinin elinde bir adet foto-anahtar olduğunu varsayıyoruz. Elde ettiğimiz gizlenmiş fotoğrafı ve foto-anahtarı SIFT’ten geçiriyoruz ve eşleşen noktalar bize **ana fotoğraftaki** odaklanmamız gereken noktaları veriyor. Burada ana fotoğraf ve gizlenmiş fotoğrafın aynı boyutlarda olması önem arz ediyor.

1. Technical Work

* 2-4 pages: Detailed explanation of your work. Provide all necessary technical details: Algorithms, APIs, programming languages, all design and implementation details…

Fotoğrafların Ayarlanması

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

OpenCV ile fotoğrafların grayscale 8 bit depth ile okunması. Burada “secret” gizlenecek fotoğraf, “img” içine gizleneceği fotoğraf, “img2” ise bahsettiğim foto-anahtar.

metin, portakal, karanlık içeren bir resim

Açıklama otomatik olarak oluşturuldu

Burada fotoğrafların birleştirilmesi işini yapıp daha sonra ondan gizlenmiş fotoğrafı ayrıştırıyoruz. Hikâyeye uygun olması açısından elimizde orijinal fotoğrafların olmadığını varsayıyoruz.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

encrypt\_image\_inside\_image: fonksiyonunda bize verilen image dizisinin ve secret dizisinin elemanlarını geziyoruz bu sırada ikisinin de elemanlarını binary’e çeviriyoruz.

Örneğin image[i,j] = 145 ise 10010001, secret[i,j] = 212 ise 11010100 olur.

Sonrasında image dizisinden gelen binary sayının en anlamlı 5 bitini, secret dizisinden gelen binary sayının en anlamlı 3 bitini birleştiriyoruz ve bu binary sayıyı tekrardan 10’luk tabana çeviriyoruz ve bu bizim şifrelenmiş yeni fotoğrafımızın bir elemanın oluşturuyor.

Örneğin image = **10010**001, secret = **110**10100 ise output[i,j] = 150 yani **10010110** olur.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

decrypt\_image\_inside\_image: fonksiyonu verilen image dizisinin elemanlarını binary sayıya çevirir, son 3 bitini alır, sonuna 5 tane 0 bit ekler ve elde ettiği diziyi döner.

Örneğin image[i,j] = 150 yani 10010**110**

ise output[i,j] = 192 yani **110**00000olur. Görüldüğü gibi secret[i,j] = 212 ile decrypt edilmiş output[i,j]=192 arasında 20 fark var yani tamamen kayıpsız bir şifreleme yapmıyoruz.



metin içeren bir resim

Açıklama otomatik olarak oluşturulduAyrıca az önce bahsettiğimiz yöntemle sadece piksellerin en anlamsız bitlerini değiştirerek yazı saklaması yapıyoruz.

encrypt: fonksiyonunda verilen s yazısını verilen arr dizisinin en önemsiz bitlerine gömüyoruz. Her s yazısındaki her char ASCII değeri olarak arr dizisinin ardışık 8 elemanının son 8 bitinde saklanıyor. En son ise kullanılmayacağı varsayılarak “\*” (42) operatörü ile şifrelemenin bittiği gömülüyor.

Örneğin s = “a”, ASCII a = 97, 97 = **110**0001

arr = [151,152,153,..,170]

arr[0] = 151 , 10010111 new\_arr[0] = 151, 1001011**1**

arr[1] = 152 , 10011000 new\_arr[1] = 153, 1001100**1**

arr[2] = 153, 10011001 new\_arr[2] = 152, 1001100**0**

metin içeren bir resim

Açıklama otomatik olarak oluşturuldudecrypt: fonksiyonunda verilen arr dizisinin elemanları ardışık olarak 8’er okunur ve en anlamsız bitleri birleştirilir, onluk tabana çevrilir ve bu sayıdan ASCII tablosundan şifrelenmiş mesaj oluşturulur.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu Sift: sırayıla k, sigma, octave, ve scale değişkenleri alıyor ve bir SIFT nesnesi oluşturuyor. Buradaki k değeri makaledeki √2, sigma değeri ise makaledeki (√2)/2 değeri seçildi, octave ve scale değeri bu değerlerden hesaplanabiliyor ama bu değerler için makalede hesaplandığı için hesaplamakla uğraşmadım. Ayrıca karşılaştırmak için openCV’nin SIFT’ini oluşturuyorum.

Burada decrypt ettiğimiz fotoğrafın SIFT’i ile foto-anahtar olan img2’nin SIFT’ini çıkarıyoruz. Ve descriptorlar yardımı ile eşleşmeleri buluyoruz fakat bu eşleşmeleri orijinal fotoğraf olan img üstünde gösteriyoruz böylece orijinal fotoğraf üzerinde gösterilmek istenen bölgeyi tespit etmiş oluyoruz.

Kendi yazdığım Gaussian blur yapan class. Fakat yazdığım SIFT’te bunu kullanmadım çünkü daha kötü bir sonuç elde ediyordum.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldumetin içeren bir resim

Açıklama otomatik olarak oluşturuldugaussianFilter: Fonksiyonda önce Gaussian kerneli hesaplıyoruz ve yuvarlıyoruz en doğru sonucu virgülden sonraki 4 basamağı alarak aldım. Gaussian Kernel seperable olduğu için bunu fotoğrafa openCV’nin sepFilter2D fonksiyonu ile uyguluyoruz simetrik olduğu için X ve Y eksenlerinde aynı kernelden geçiriyoruz. Ve blurlanmış outputu dönüyoruz

metin içeren bir resim

Açıklama otomatik olarak oluşturuldugaussian: openCV’nin kullandığı gaussian değer formülü o r değerinin gaussian değerini döner.

gaussianKernel: verilen sigma değerinin gaussian kernelini oluşturur. Toplamı 1 olacak şekilde kerneli döner.

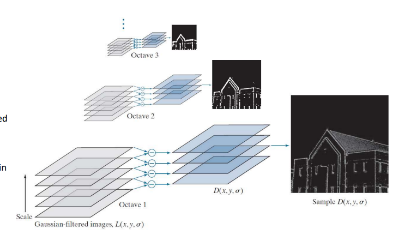
SIFT

metin içeren bir resim

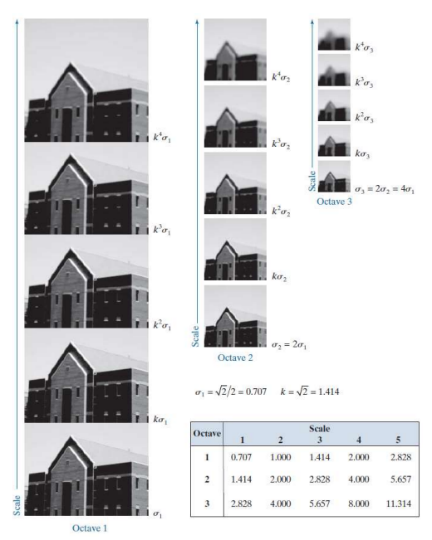
Açıklama otomatik olarak oluşturulduOpenCV’nin KeyPoint classında scale ve layer tutmadığım için kendi keypoint classımı yazdım. Ekstra bir özelliği veya fonksiyonu yok.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldudetectAndCompute: Scale space ve DoG space’i oluşturur. DoG space’i kullanarak maksimum ve minimum noktalardan keypoint çıkarımını yapar. Ünik olmayan keypointleri elemine eder, boyutunu dönüştürür. Ve bu keypointlerden descriptorları çıkarır. En son myKeyPoint classının keypointlerini döner.

metin, ekran görüntüsü, ekran içeren bir resim

Açıklama otomatik olarak oluşturuldu

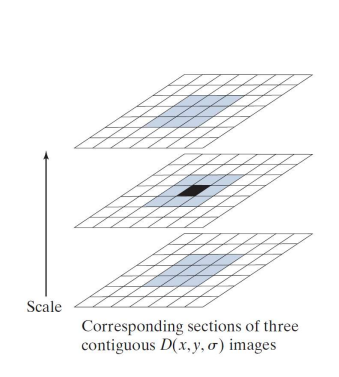
metin içeren bir resim

Açıklama otomatik olarak oluşturulducreateDog: scaleSpace’deki fotoğrafların arasındaki farkı çıkarır. Burada elde ettiğimiz DoG image, LoG image’e çok yakın değerler ürettiği için ve LoG çıkarmak burada masraflı olacağı için tercih ediliyor.

createScaleSpace: aldığı fotoğraftan (octave x scale) büyüklüğünde bir gaussian image space’i oluşturur. Burada başta yaptığımız upsampling daha fazla detay olduğundan daha fazla keypoint elde edebilmemizi sağlıyor. Makaledeki sigma değişikliklerini uygulayarak her seferinde fotoğrafı 2x küçülterek blurluyor.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

createMaxMinSub: Burada amacımız 26 komşuyu kullanarak merkezin maksimum veya minimum olup olmadığını kontrol etmek. Komşular dogSpace[i,j-1], dogSpace[i,j+1] ve dogSpace[i,j] (merkez dahil değil) olup, ana makalede yapıldığı gibi lokalizasyon yapılması gerekli çünkü sürekli bir fonksiyon yakalandıysa daha iyi bir lokal maksimum ya da minimum bulunabilir.

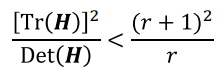
metin, iç mekan, ekran görüntüsü, kağıt içeren bir resim

Açıklama otomatik olarak oluşturuldumetin içeren bir resim

Açıklama otomatik olarak oluşturuldumetin içeren bir resim

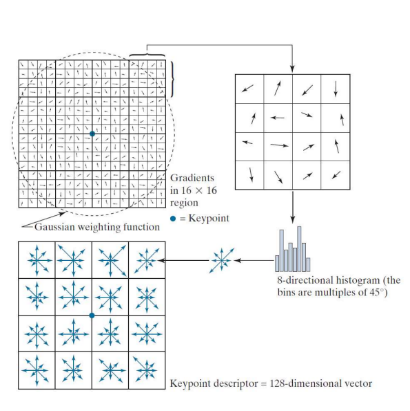
Açıklama otomatik olarak oluşturuldu

metin içeren bir resim

Açıklama otomatik olarak oluşturuldumetin içeren bir resim

Açıklama otomatik olarak oluşturuldumetin içeren bir resim

Açıklama otomatik olarak oluşturuldulocalizationQuadratic: Burada amacımız en iyi lokal maksimum veya minimum bulmak. Bunu yaparken de gradiyent ve hessian matristen yararlanıyoruz. Bunu en fazla 5 kere tekrar ediyoruz. Eğer 3 yönde de 0.5’ten az değişiyorsa bir lokal maksimum veya minimum bulmuşuzdur döngüyü sonlandırıyoruz. Eğer makalede belirtilen 5 döngü sınırına ulaştıysak ve hâlâ bulamadıysak lokal maksimum bu pencerenin içerisinde değildir diyoruz ve None type dönüyoruz. Eğer bir lokal ekstremum bulduysak bunun low contrast bir nokta olmadığına karar verirsek kenar, düzlük olup olmadığını kontrol ediyoruz. Köşe ve düzlük olup olmadığına oluşturduğumuz hessian matristen yararlanarak karar veriyoruz. Kenar veya flat bir alan değil ise keypoint’i oluşturuyor ve dönüyoruz.

keypointOriantations: Burada amaç döndürmeden bağımsızlık kazanmak. Bunun için basitçe 360 derece için 10’ar derece ile 36 tane parçası olan bir histogram oluşturuyoruz. Her derece için gaussian imageleri kullanarak gradient magnitude hesaplıyoruz. Makaleye göre scale 1.5 kat olmalı. Ve 0.8’den yüksek peak’e sahip olan parçaları alıyoruz ve bu keypointe oriantationı’nı da ekleyip saklıyoruz.

generateDescriptors: 16x16 komşuları alıyor bunu 4x4 16 bloğa ayırıyoruz. Her parça için 8 parçası olan oryantasyon histogramı oluşturuyoruz. Yani 8x16=128 parçalı descriptor vektörümüzü elde ediyoruz. Sonrasında ağırlıklı gradient magnitude’u yumuşatmak için trilinear interpolasyon uyguluyoruz. Daha sonra bir threshold’dan geçiriyor ve normalize ediyoruz. (Bu parça tamamen internetten aldığım tek parça)

1. Experiments and Results

* 1-3 pages: Detailed explanation of your experimental setup and results. Provide all necessary technical details: Dataset, hyperparameters, visual or other results…

metin, bina, açık hava, ev içeren bir resim

Açıklama otomatik olarak oluşturuldumetin, kişi, grup, insanlar içeren bir resim

Açıklama otomatik olarak oluşturuldu

Orijinal Fotoğraf Gizlenecek Fotoğraf

metin, kişi, grup, insanlar içeren bir resim

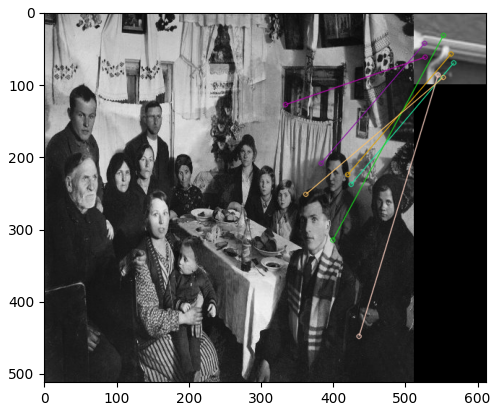
Açıklama otomatik olarak oluşturuldumetin, açık hava, pencere içeren bir resim

Açıklama otomatik olarak oluşturuldu

Orijinal Fotoğrafın içine secret adlı fotoğrafın Şifrelenmiş fotoğraftan gizlenmiş fotoğrafın

gizlenmiş hali gözle görülür bir fark yok. tekrar elde edilmiş hali gözle görülür bir bozulma var.

Orijinal fotoğrafı deşifre edecek kişide bulunduğunu varsaydığımız foto-anahtar.



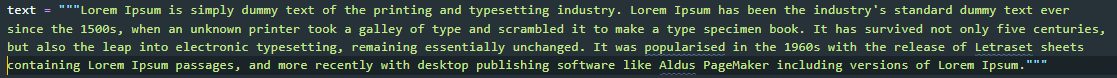


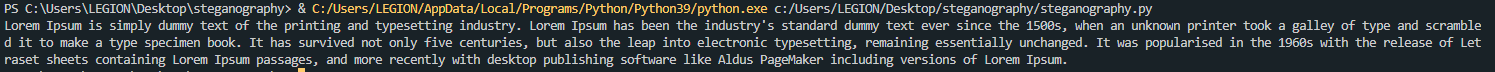
Benim yazmış olduğum SIFT classının elde ettiği sonuçlar. Eğer deşifre edecek kişi aynı örüntüleri takip eder ve etrafını anahtar boyutunda bir çerçeveye alırsa hedefi açıkça belirleyebilir. Kırmızı kalemle çizdiğim şekilde.



metin, grup içeren bir resim

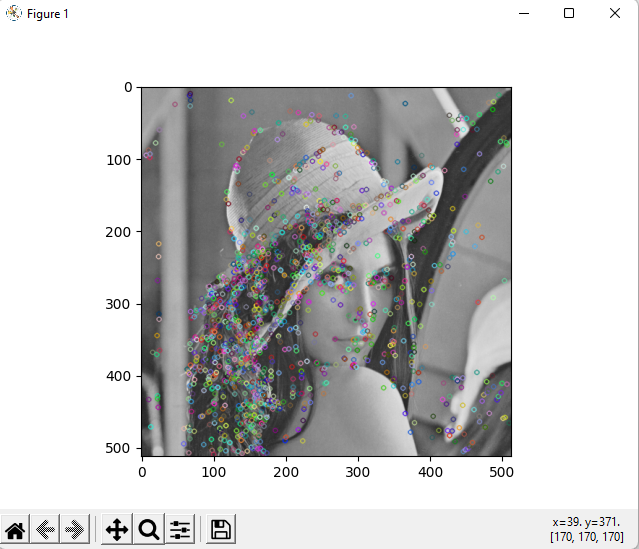
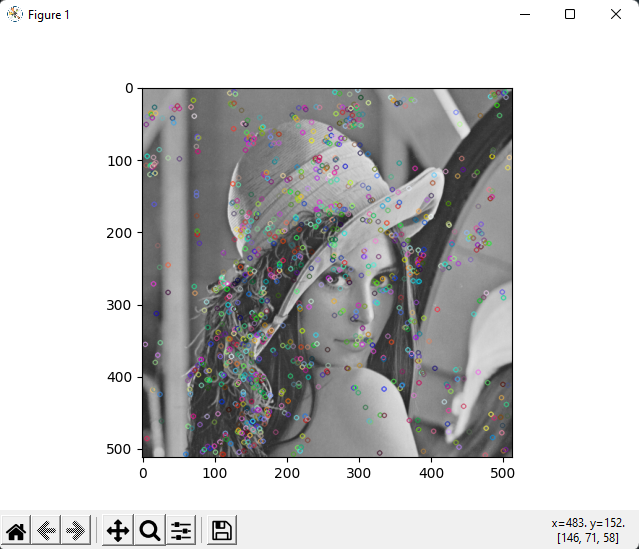
Açıklama otomatik olarak oluşturuldu

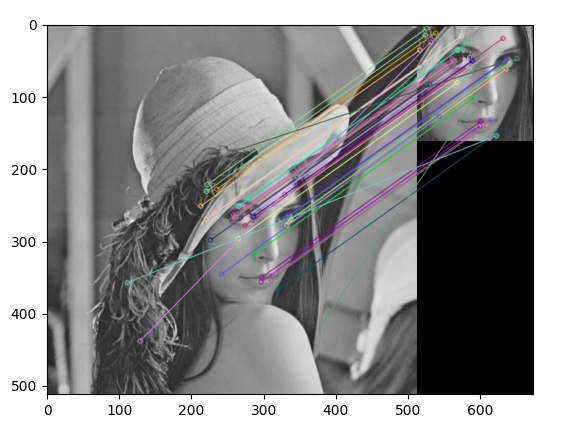
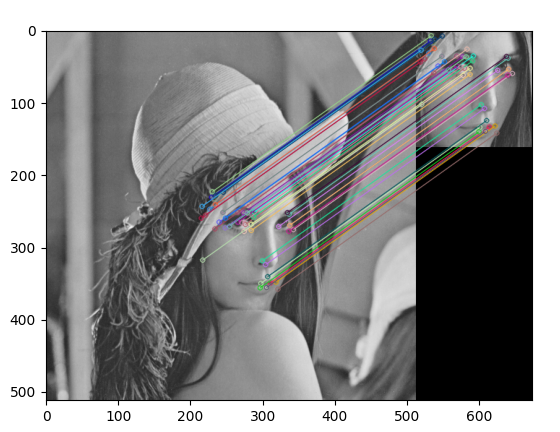
Opencv’nin SIFT’ini kullandığım zaman çok daha iyi bir eşleşme elde ediyorum. Hedef açık bir şekilde belli oluyor. 

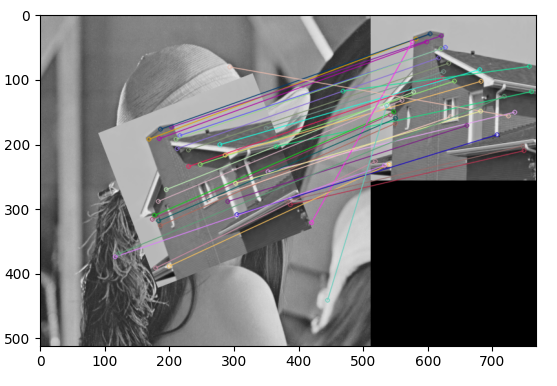
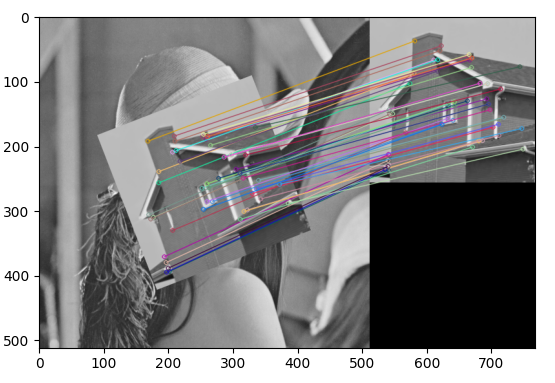


Orijinal fotoğrafın içine gömülmüş text’e aynen ulaşıyoruz.

SIFT DENEMELERI

↓ ↓ ↓ Benim SIFT keypointlerim ↓ ↓ ↓ ↓ ↓ ↓ OpenCV SIFT keypointleri ↓ ↓ ↓

 ↓ ↓ ↓ Benim SIFT eşleşmelerim ↓ ↓ ↓ ↓ ↓ ↓ OpenCV SIFT eşleşmeleri ↓ ↓ ↓



1. Conclusions

* One paragraph: Briefly explain your observations and what you have learned.

Benim asıl amacım fotoğraf içerisine fotoğraf gizlenmesiydi fakat bunun çok basit olduğunu gördüm ve şifreleme çeşitlerine girmenin dersin amacından uzaklaşacağını düşündüğüm için aklıma gelen bir fikri koda dökmek istedim. SIFT’I implemente etmek beni çok zorladı fakat bunun aslında lineer cebirden başka bir şey olmadığını gördüm. Bunu fark ettikten sonra formülleri anlamam daha kolaylaştı. SIFT’i yazarken ne kadar ortada bir makale olsa da OpenCV yazarlarının birçok noktada optimizasyonlar yaptıklarını gördüm olabildiğince bunları ben de aktarmaya çalıştım. Ayrıca OpenCV’nin işlemleri yapma şekli (multi-threading vs.) açık bir şekilde hız farkı doğuruyor. Bu proje sanırım şimdiye kadar yazdığım en derin matematik içeren kodumu yazmamı sağladı. Ayrıca ilk defa bir makalenin kodunu yazmak da bana ileride yayınlayabileceğim herhangi bir makalede neleri göz önüne almam gerektiğini gösterdi. Ayrıca bu proje debugging yeteneğimi inanılmaz geliştirdi.